Software Defined Network Security Framework for IoT based Smart Home and City Applications

Song Wang, Karina Mabell Gomez, Kandeepan Sithamparanathan and Paul Zanna*
School of Engineering, RMIT University, Melbourne, Australia,
*Northbound Networks, Melbourne Australia
Email: {name.surname}@rmit.edu.au and paul@northboundnetworks.com

***Abstract*—As a popular application of Internet of Things (IoT), Smart City Frameworks aim to provide real time tracking, intelligent control and surveillance across the city. Thus the improvement of resource utilization is a big concern in the management, how to administer such a massive network to meet the requirement of different services? Software Defined Network (SDN) is an ideal solution in customizing networks; however the security feature is the common challenge in both SDN and IoT. In this paper, we propose a framework that uses smart techniques for improving the security features of SDN for smart city applications and diminishing the risk of network invasion. Our SDN Security Framework (SDN-SF) combines two techniques: i) it restrains the unnecessary path between IoT nodes, and ii) it classifies devices into three levels from a combination of MAC address and HTTP request. Additionally, thresholds derived from historical behavior are used for anomaly detection in order to enhance network adaptation. Our result collected from real SDN-based IoT testbed demonstrates that our SDN-SF for Smart City scenarios is able to detect and mitigate malicious traffic with 99.9% of detection rate and 0.5-1 second of detection time in both the control and data plane, respectively.**

***Index Terms*—SDN, Security, IoT, Smart City, Network Attacks, Anomaly Detection.**

I. Introduction

IoT has received high attention due to its usage across different fields. IoT is composed of large and dense networks providing routes to achieve real time transmissions from anywhere. IoT utilizes intelligent process to control and manage the enormous amount of data collected from sensor, devices and users. IoT applications range from a portable device for health monitor to an entire procedure of urban operation and maintenance on industrial applications. Some IoT applications gather feedback from customer behaviors so as to help vendors to improve the product or service, such as smart retail. Others IoT applications work 24/7 to keep an eye on a small area or component, such as smart agriculture applications. Also IoT applications are used to upgrade the quality of life, such as smart home [1], [2]. Although the majority of IoT scenarios are in the wireless environment, it is just data collection on the access layer, the big granular data delivery in the convergence or backbone network are still wired communication because of the stability and reliability.

Since IoT involves billions of devices, from a sensor to a server, it brings new threat along with its advantages [3], thus traditional network could hardly handle the management [4] of such networks. While SDN has more flexibility and programmability than the legacy network, because it decouples the control plane (controller) from the data plane (switch) to achieve centralized management. It is able to modify data routes in real time without manual intervention for both normal transmission and security reasons [5]. The applications running over the controller are developed refer to the customer or individual requirement, it is an open source platform that people are encouraged to cover as many areas as they need without worrying about the cost of license. Its programmability satisfies the diverse quality of service (QoS) demand of IoT applications under the same network. Even if there is any update of policies, a modification in the application is enough to upgrade the whole network [6]. Thus, SDN becomes an ideal substitution to administer an IoT network.

In a small IoT network, such as smart home, communication within the same domain are generally wide open to all the devices sharing the same wifi network [7] or small LoRa network. While in a large IoT network, such as smart city, most of the components are pre-deployed, and these units may have paths to the kernel [8]. However, some of the paths between IoT devices are not required to exist in the network since no interaction are required. Thus, to interrupt unnecessary path in the smart city is an enhancement to the security level in the IoT network. In this paper, we propose a novel interactive framework that uses smart techniques for improving the security features of SDN for smart city applications and diminishing the risk of network invasion, our results show that our technique is able to detect and mitigate malicious traffic with 99.9% of detection rate and 0.5-1 second of detection time. First, our SDN-SF dynamically constrains the communication route of IoT devices to avoid unneeded path connection, so that malicious users have less opportunity, as well as lower significance, to compromise a device. Then, the SDN-SF learns the behavior of each device to create a dynamic threshold that reflects the regular actions of a single device. After, the SDN-SF classifies the IoT devices into different categories according to the device type, and assigns access permission to the relevant category. Finally, the SDN-SF involves administrator intervention to authenticate new devices for the sake of security in the core network area.

The rest of this paper is organized as follows. Section II reviews relevant works. Section III describes the proposed system model while Section IV introduces the algorithm. Real implementation of the algorithm is illustrated in Section V. Finally, results discussion and conclusions are given in Section VI and Section VII, respectively.

978-1-7281-2194-9/19/$31.00 ©2019 IEEE

II. Related Work

As two emerging network trends in the near future, IoT and SDN are being merged to make up the deficiency of each other. Xu *et al.* [9] proposes to draw on the architecture of SDN to build a software defined smart home (SDSH) platform. It consists of three layers: external service layer, controller layer, and smart device layer. SDSH finds the location of user at home and collects environment status, from brightness to food storage, around him via smart devices, in addition to the trigger of home appliance and reminder messages, the artificial intelligence (AI) in the SDSH keeps learning user habit to improve the service, and the employment of SDN produces customization of smart home. However, SDSH is only a theoretical model and a main challenge is the security of privacy information. Flauzac *et al.* [10] introduces a multi-domain SDN architecture to authenticate IoT devices, it uses controllers, called Border Controller (BC), deployed at the edge of each network domain to achieve inter-domain communication. Each IoT device needs to obtain access permission from the controller to join a network domain. Once the destination of a request is not on the same domain, the relevant BC will inquire neighbor BCs to find the route and then update routing table on all the switches along the way. Although, it tries to enhance IoT security via device authentication, detailed process is not mentioned and no test/simulation has been done.

Xu *et al.* [11] designed a smart security mechanism to protect SDN-based IoT against new-flow attacks. In the detection stage, it defines request rate and matches efficiency to differentiate a new-flow attack from a normal flow burst. For mitigation phase, it uses a security middle-ware to filter malicious packets, and informs the controller of the result so as to update flow entries on the victim switch. Time slot of each monitor period is 10 seconds, and mitigation normally requires 3 slots, which means it needs 30s to react. From the simulation, the precision rate is about 86% which still has room for improvement. El-Mougy *et al.* [12] described several existing SDN-based IoT solutions mainly focus on the network management, the main idea is to manipulate controller as a real time implementer. As long as an operator or user has a request, the controller is then responsible for establishing a route to provide desired services. Particularly in the wireless sensor network, it is necessary to take energy efficiency into account for the controller when calculate the best route. The centralization of SDN controller creates flexibility to adapt to IoT network. In the meanwhile, the synchronization between multiple controllers becomes a new challenge. Moreover, the security and privacy are important but not easy to solve.

Vilalta *et al.* [13] illustrated a SDN-based frame to filter IoT malicious data, this frame engages a collector, a detection module and a mitigation module running over the SDN controller. Collector gathers data in order to form statistics, which will be applied to be the reference of the anomaly detection module. The detection module will inspect flow table on the SDN switch at a regular interval, attempting to identify a potential attacker. Based on the decision in the detection module, traffic is forwarded to the destination or blocked through the mitigation module. Kalkan *et al.* [14] associated the SDN controller into three roles: intrusion controller, key controller, and crypto controller. Intrusion controller is responsible for attack detection and routing calculation; key controller keeps and assigns the key to the IoT device, while crypto controller provides cryptographic services. This architecture aims to distribute the risk of controller failure to three different segments, however the authors did not mention what if the intrusion controller is down, who is going to compute a route. Bhunia *et al.* [15] presented a SDN-based framework, called SoftThings, with Machine Learning running on the controller to study the behavior of IoT devices. This framework employs cluster controller (CC) who is only in charge of a small group of IoT devices, and a master controller (MC) is working on the highest level with an overview of the entire network to manage those CCs. A CC observes traffic pattern and classifies traffic with Machine Learning, it keeps the MC updated on the route information in its own cluster.

From the aforementioned solutions, real implementation is rarely performed, and a few transmission routes in the IoT are actually not required under normal circumstances for both management and security. Thus, the combination of proactive and reactive SDN based network will constrain the spread of infection and conserve network resources since it prohibits the useless path.

III. SDN Security Framework System Model

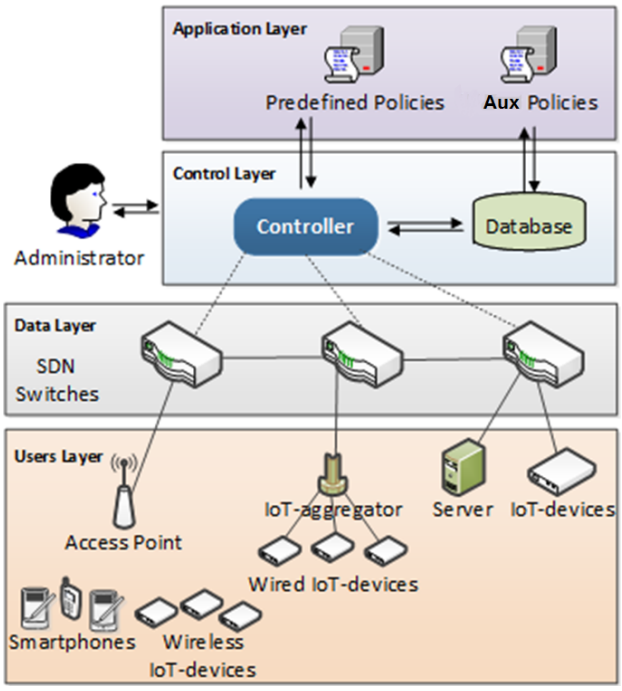


Fig. 1: SDN Security Framework (SDN-SF) system model architecture composed by Application, Control, Data and User Layers.

The SDN Security Framework (SDN-SF) system model is depicted in Figure 1. In the *application layer*, administrators

prepare both proactive (predefined policies) and reactive (auxiliary policies) rules to manage the network.

- **The predefined polices** define the routine behaviors in the network during initialization, as well as add new or replace current devices, these policies mainly concentrate on the normal connection between network devices regardless of any threat. Predefined policies are sent to the SDN switch via controller, any modification of device's physical information, such as MAC address, shall be updated by the administrator. For example, a device needs to be replaced due to a fault, then the MAC address of the new device shall be added in the predefined policy, in the meantime, the MAC address of the old device shall be erased.
- **The auxiliary policies** are responsible for collecting statistics from and creating thresholds on the control and data layer. It generates dynamic flow rules to manage real-time network behavior, as well as the threshold for anomaly detection. Auxiliary policy defines the method to create thresholds based on the information collected via the control layer, and new thresholds are stored in the database. Statistic in the database of control layer assist to make the threshold, once the excess of threshold happens, the auxiliary policy will insert flow entries with higher priority to the SDN switch via controller to modify the transmission of malicious flows. The auxiliary policy is used to implement the security feature in the SDN-SF.

In the *control layer*, the controller manages the entire network, from routing to monitoring. It puts all the statistic of the network to the database, calculates threshold for each device at each time slot according to the guideline given by auxiliary policy. The controller also compares real-time counter with threshold to determine if the network is safe. If it is unsafe, the controller will update the flow entry on the SDN switch to change relevant forwarding path. The database here acts as an interchange station of the network behavior and routing policy, it provides a reference of the current network circumstance.

SDN switches in the *data layer* enforce the command from controller, they may redirect traffic to the destination, or to a filter for further examination, or even block the traffic if suspicious. *User layer* includes IoT devices and aggregator, they generate traffic and send it to server via SDN switches. Traffic initiated from wireless/wired devices, like cell phone, is sent to an aggregator first, from the aggregator/access point it is then transmitted to the SDN switch. The SDN-SF mainly focuses on the behavior on the application, control and data layer, which could be regarded as the core network in the IoT traffic transmission.

A. IoT-devices Categorization

Since traditional network can only forward packets to the destination, further examination is processed on the end device, thus the advantage of running IoT over SDN is the ability to distinguish different devices and to learn their behaviors. If the end device is not smart enough, such as a typical IoT sensor, it is easy to be compromised by an attacker sharing the same network. In the proposed SDN-SF the communication between devices will be filtered before transmission, the SDN-SF API running on the controller recognizes the device via its User-Agent (UA) string in the HTTP header (for large vendors with a wide range of products, such as Samsung), or its MAC address (for small vendors with limited products, such as Raspberry Pi). The UA string identifies the version of software and hardware running on the device, so that it is able to categorize IoT end devices into several groups. For the device with no application layer, unable to run HTTP, routing path will be created only with its IoT-aggregator. Thus the SDN-SF divides IoT devices into three categories:

- *Basic IoT-devices:* Devices, such as sensors on the light, only have access to the next level device, such as an IoT-aggregator or data collector, basically these devices do not have peers and application layer. Since the communication paths to other basic devices are restricted, it increases the workload for hackers to compromise multiple bots.
- *Enhanced IoT-devices:* Devices can be identified by MAC address, the behavior is predictable, similar to the basic IoT-device. Normally, enhanced IoT-devices are installed or placed in the specific areas without mobility. They are able to control and manage basic devices, using command given by the users, or implementing applications to the basic devices. Its action is recorded and studied over time so that anomalous behaviors are detected.
- *Advanced IoT-devices:* Smartphones are typical advanced IoT-devices, which have access to all the basic and advanced IoT-devices. These devices have mobility and unpredictable behaviors, so approval of administrator for authentication is required. These devices are capable of retrieving real time information, receiving reminders and making decisions to update the configuration of IoT network or other IoT-devices.

The IoT-device recognition improves the network security, because majority of IoT-devices are basic, so easy to be compromised physically or logically due to its broad-coverage deployment. Thus SDN-SF narrows or fixes their communication paths while suspicious communication paths need authorization or will be blocked.

B. IoT-devices Registration

In our SDN-SF the database stores MAC address, device type, as well as a potential device list, such as various brands of mobile phone. Devices in the IoT network can also be split into public and private ones. The *public devices* normally have no confidential or privacy data, i.e. access point is a typical public device. While the *private device* requires admission from administrator, i.e. PC is a classic private device. A device connected to the IoT network shall firstly adopt HTTP message or identifiable MAC address to register itself as an enhanced or advanced device. From the existing database of device type, a relevant predefined access control list starts to work, and MAC address will be used to distinguish the source of requests. From the information in the database, the SDN

controller will judge whether the request is to enter a private device without authorization, if so the request is dropped and a reminder is sent to the administrator with the device type. If the unauthorized device tries to access again, there are two possibilities i) it is detected as miss-operation in that case no further message is sent to the administrator, or ii) it is detected as DDoS attack in that case this is notified to the administrator.

In the database, existing devices are already registered and mapped to relevant device types. However, a new device needs to access a website to register before it is allowed to connect, which is similar to the connection to a public wifi. Except for basic devices, they are generally deployed refer to a pre-made plan or replaced due to any fault, thus, a new basic device shall join the network with the approval of the administrator. The approval is just the update in the database, put the new MAC address in the list and no more operation.

C. IoT-devices Management

The predefined policies are applied in order to manage existing devices in the network, these devices are trustworthy at the beginning as those are manually installed and gain the authorization to join the IoT network from administrators. Predefined policies just define the original access control list and planned periodical transmission of a few devices, detailed real-time network operations are controlled by the auxiliary policies. Whether a packet is going to pass a filter is conducted by the auxiliary policy according to the history log, even the earliest added devices may be urged to use the filter, because many IoT devices are placed in the public area without 24/7 surveillance, it is possible to be replaced with a malicious device having the same tampered MAC address and other hardware/software information.

Initially, predefined policies generate flow entries in the SDN switches to run the network, known devices are able to work under a proactive circumstances with their behaviors recorded in the database. And from the normal behavior, the database will create a auxiliary policy for each device, this policy sets a dynamic threshold for the specific device. As long as the behavior is abnormal, the auxiliary policy adds a flow entry to forward packets to the filter first, and only genuine packets will be sent to the destination. If all the requests are legitimate, then the database is going to update the history log and auxiliary policy. Malicious request is blocked in the filter otherwise. Here, the predefined policy determines whether a device has access, while the auxiliary policy determines if the device is working suspiciously, so there is no conflict between these two policies. Furthermore, for the flow rules they set up in the SDN switch, the auxiliary policy will use a higher priority than predefined one's so that the traffic will be delivered to the destination via a filter until the auxiliary flow rule is removed.

IV. Proposed Security Model

The administrator could define a sensitive area that contains critical equipment, such as core servers and switches, the access to this area is strictly constrained, as any misoperation or malicious attempt could result to network damages. Thus, only the devices with permission are allowed to enter the sensitive area, otherwise they only have access to the public area. Additionally, the proactive routes are prepared when the application begins to work, relevant flow entries are sent to the switches without any trigger. Besides, the administrator needs to input device information, such as IP address or MAC address, in the application first. And those entries are preferred than new created entry, they will not be removed unless manual intervention due to equipment replacement. If there is an interruption in the network, the controller will find another route and give a higher priority to forward traffic before the issue is fixed.

A. Device Registration

A database in the controller already has Organizationally Unique Identifier (OUI), device type and device level for known devices. The working procedure of a known device is depicted in the Figure 2. After it begins to work, the controller starts to monitor and compare its behavior with the policy to define if it is real traffic. The policy is a dynamic threshold that changes over time, it is generated from the continual study in the history log for anomaly detection. Once the behavior of a device is abnormal according to its threshold, a new flow entry, however, will send these traffic to a filter to guarantee the transmission of legitimate users. If all of them are normal, or the number of normal flows exceeds threshold, then the policy is updated to adapt to the new environment. Through the record-and-compare mechanism, the policy will improve itself to service the network.

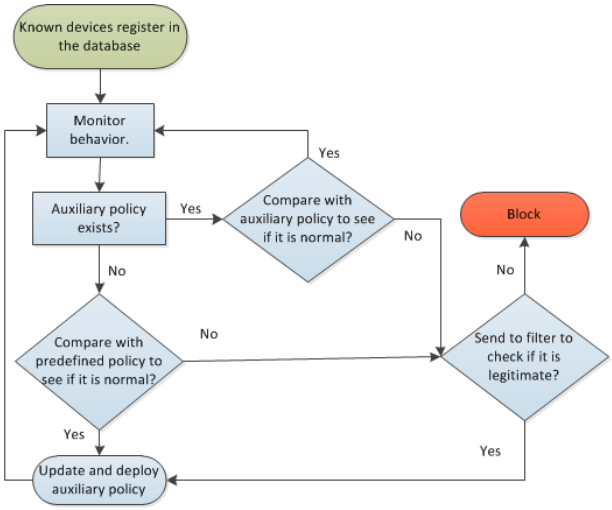


Fig. 2: Procedure of known devices working in the network.

While for a new device trying to connect to the IoT network, the user shall access to a website first, by which the controller will extract the operating system (OS) of device from its HTTP request. Its MAC address, as well as OUI, is recorded in the database, and the combination of OS and vendor from OUI could expose the character of the newcomer. Then the controller will try to match the device level in the database, and consult administrator if it is authorized to involve manual

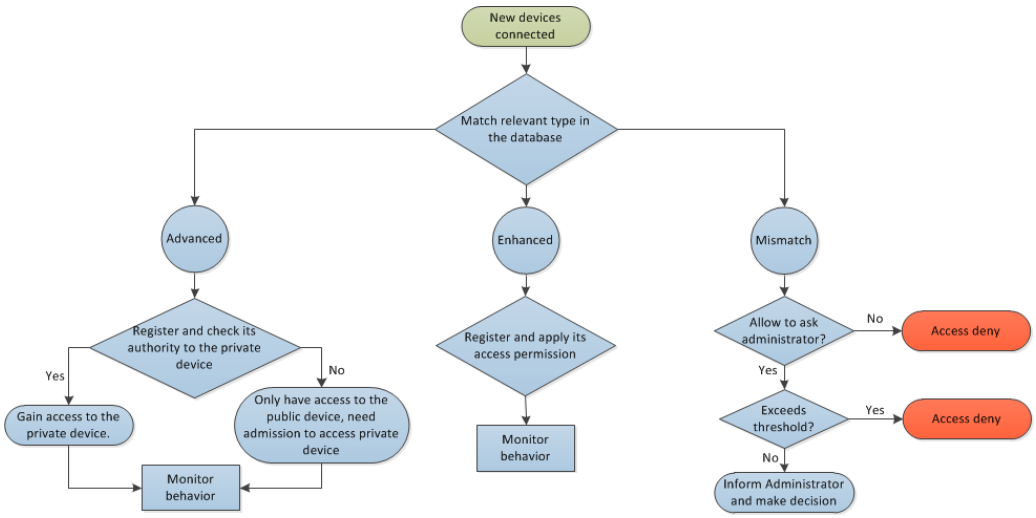


Fig. 3: Procedure of new devices working in the network.

operation when mismatch. In case of denial of service (DoS) attack, number of inquiries to the administrator is limited to a low value, because the database stores all the OUI information of current vendors, the probability of a mismatch is minimum. It should be noted that registration is not required for basic devices, because these IoT components are normally layer 2 or layer 3 devices, which are deployed in advance. Their connection shall not be arbitrary.

B. Network Monitor

In order to create a dynamic threshold which could reflect the real time network utilization, collection of as many samples as possible is an effective way to study the behavior of a device during a long period. The number of sample slice depends on the requirement of application and customer. Accumulation of the statistic produces a dynamic sophisticated threshold at each time slot, and the excess triggers the traffic to pass through the filter so as to achieve early detection. If a statistical period is α, and it is required to split into n pieces, the monitor period β then equals to α/n. Each β has a dedicated counter for a specific device to collect information, so that for a single device, there are n results after α. Initially, a default group of threshold is prepared to provide a fundamental reference. Then we take the mean of history record to create a new threshold. The effect of this default threshold is neutralized along with the build-up of statistics after a long time. From this sampling method, it is able to illustrate a graph of the real time threshold for each IoT device. Both the control plane and data plane are under surveillance, yet the object and measure of data collection are different.

- Control plane (controller): As a vulnerable single point-of-failure node, the controller runs a group of Packet_In message counters to track the behavior between the controller and switch. Packet_In message is a request from the switch to controller for action consultation, it is an effective way for a host to contact controller indirectly [16].
- Data plane (switch): A switch keeps statistic of each flow entry, which includes ingress/egress, byte count. This statistic only contains performance in the data plane, and the controller is able to collect these data from the switch.

C. Anomaly Detection

Detection of suspicious behavior is based on the comparison with threshold, moreover, control and data plane have different detection method.

- Control plane: As the threshold for current monitor period is known, the controller will measure the duration γ from the first Packet_In comes till the counter hits the threshold. If $\gamma < \beta$, the behavior is abnormal and packets from the malicious source MAC address will be forwarded to the filter.
- Data plane: Since the controller will receive byte count report only after it requests from switch, the audit has to be done per β. The controller investigates the byte count of each flow entry, if the number is greater than threshold, relevant traffic flow will be sent to the filter.

Through the filter, real traffic is sent back to the network, and the controller is capable of requesting for the port statistic from the switch which is directly connected to the filter. From the number of legitimate packets, the controller will update the threshold as well. As long as the traffic towards controller or switch becomes normal, the controller will adjust the flow entry back to the previous configuration.

V. Model Implementation Details

SDN-SF is implemented and evaluated using a real SDN-based testbed as shown in Figure 4 and described as following. A laptop (Ubuntu 16.04 OS with Intel i5 CPU and 8G RAM) is running Ryu controller, RPis are emulated as Basic (RPi-A), Enhanced (RPi-B) devices, filter (RPi-C) and server (RPi-D), another laptop running Windows 7 is an advanced device. All the four RPis here are known IoT-based devices, routes

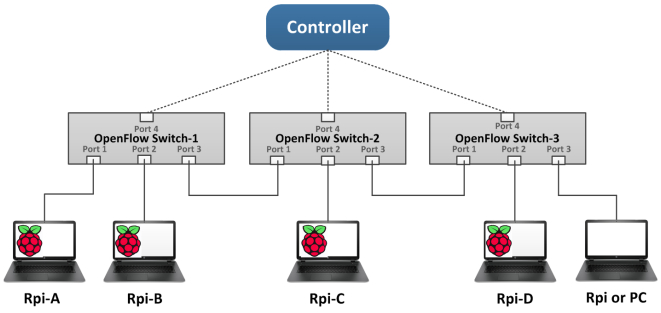


Fig. 4: SDN-based testbed for evaluating and validated the proposed SDN-SF in smart city environment.

between RPi-A and RPi-B, RPi-B and RPi-D are proactive network, however RPi-A is unable to contact server directly, it is only allowed to talk to RPi-B. Due to the device level, the controller will not pave the way from RPi-A to RPi-D at any time. RPi-D is defined as a sensitive area in this network. A new device is connected to this network via port-3 of switch-3, the controller could recognize if it is a RPi or a computer from its generated Packet_In message, and whether it is permitted to communicated with RPi-D. These operations primarily depend on the pre-defined policy in the application.

A. Traffic Generation and Monitoring

After an IoT device, known or new, starts to work, a counter associated with its switch ID, port ID and MAC address begins to record the behavior of connection to, not only the controller but also peer devices within data plane. In order to emulate random traffic in this network, we use Iperf test tool to create UDP packets with random bytes per random interval to the control plane and data plane. Both gentle and burst traffic are imitated to verify the threshold generation and anomaly detection. A test round has 30 monitor periods (n=30), and each monitor period (β) is 10 seconds, thus, a device will have 30 monitor records corresponding to each monitor period. A record in a specific time slot represents the threshold for this device within current 10-second period. For control plane, we use RPi-B to forge Packet_In request to the controller, the controller updates counters every time it receives Packet_In messages, once RPi-B's counter hits its threshold and duration is less than 10 seconds, a new flow entry is sent to the switch to forward mismatch packets from RPi-B to RPi-C. Note that RPi-C here only represents a filter but has no filter function. Hence, the controller will receive no more request from RPi-B for a while, this redirection entry is erased if no similar requests for 10 seconds. While on the data plane, we flood UDP traffic from RPi-B to RPi-D, so the controller realize the issue from the byte count statistic on the switch and redirect traffic to RPi-C. As a filter will transit legitimate packets back to the network and drop malicious packets, the controller will then obtain statistic from RPi-C to observe the status in the data plane. When the number of bytes coming out of RPi-C is lower than the threshold, RPi-B is able to reach RPi-D again. Here we stop the Iperf to ensure the end of attack due to the same reason of data rate control.

VI. Results and Discussion

In this section, we verify proactive scenarios, device recognition, and mainly concentrate on dynamic threshold under different circumstances, anomaly detection output.

After Ryu controller is launched, bidirectional flow entries are ready on all the three OpenFlow switches for Rpi-A to Rpi-B and Rpi-B to Rpi-D. In the meanwhile, Rpi-D is unreachable to Rpi-A. This flow entry deployment has been verified by accessing to the OpenFlow switch via serial port. Then, we connect a Rpi and a computer to the port 3 of OpenFlow switch-3, respectively, to verify the recognition of device. Rpi is easily recognized via its typical OUI, while a computer is registered from its User Agent string and OUI. The output for computer is given in Figure 5, User Agent indicates it is using Windows system and a Chrome explorer, in the meanwhile, its MAC address reveals the vendor, so the controller knows it is an Acer computer and categorizes it to Advanced. However, this computer is not allowed to access to the sensitive area, so it is unable to reach Rpi-D even though it is an Advanced device.

```
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: 1
ST: urn:dial-multiscreen-org:service:dial:1
USER-AGENT: Google Chrome/63.0.3239.108 Windows

Device is Acer Computer, and device level is Advanced.
```

Fig. 5: Controller output: Recognition of a new device.

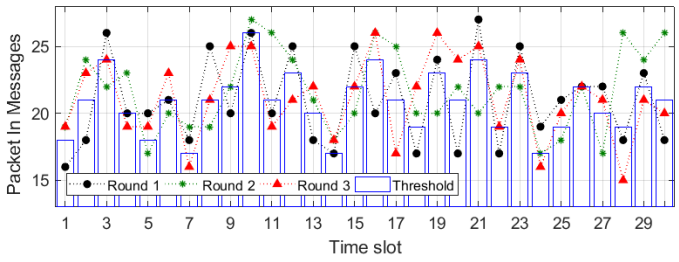
A. Control Plane Traffic Analysis

Due to the limit of number of devices in the testbed, Rpi-B falsifies Packet_In messages by sending packets to non-existent destination to emulate the request in the network. Figure 6.a shows a gentle request frequency, the threshold is stably fluctuating within a small range after 3 rounds running. If we involve burst traffic with a random and long interval to this environment, we can find in Figure 6.b the threshold at different time slot could have a huge gap.

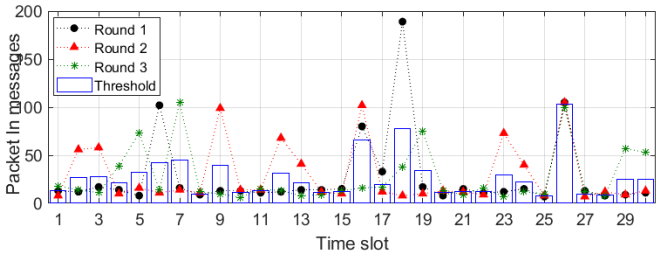
Since real-time dynamic threshold is available now, we try to engage some abnormal behavior to see the response from network. From Figure 7 can be seen that in the 5th and 22rd 10-second time slot, the number of Packet_In hits threshold, and none of the counter exceeds threshold because of the detection method. While after each of these two periods, the controller has no request from this device thanks to the flow adjustment on the switch. The counter and threshold here is dedicated to a single device, while the controller is still handling Packet_In messages from other devices. Also, we notice that 8th counter is higher than most of the threshold, but it is still normal according to the history behavior. Thus, this kind of threshold has a better adapt to the real environment.

B. Data Plane Traffic Analysis

In the data plane, threshold generation is similar to the way in control plane. Figure 8.a has only normal traffic and Figure 8.b has burst traffic from time to time, after 3 rounds test we find that byte count has greater fluctuation than control



(a) Normal Traffic.



(b) Burst Traffic.

Fig. 6: Threshold generation based on the number of Packet In messages for control after 3-round of traffic.

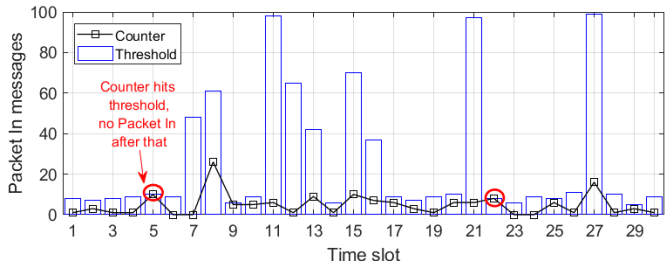
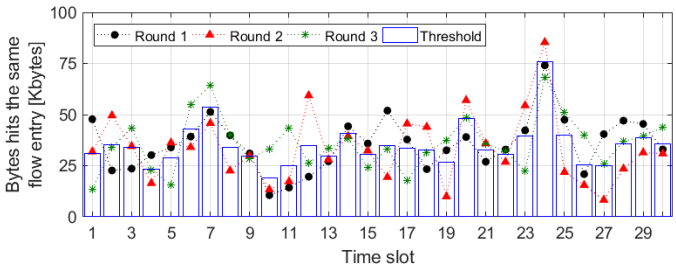
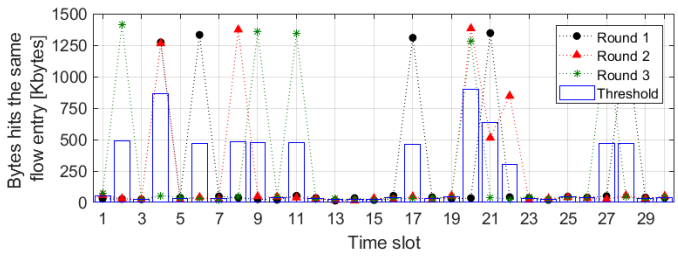


Fig. 7: Traffic behavior when anomaly behavior in the number of Packet In is detected on the control plane.



(a) Normal Traffic.



(b) Burst Traffic.

Fig. 8: Threshold generation based on number of bytes on the same flow entry for data plane after 3-round of traffic.

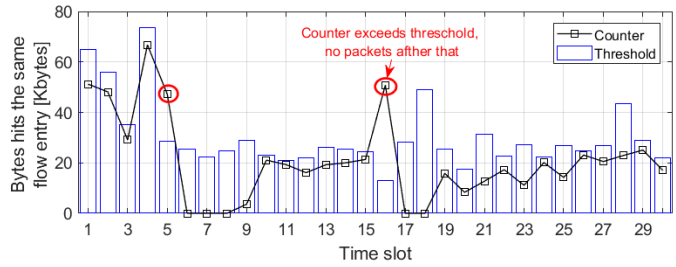


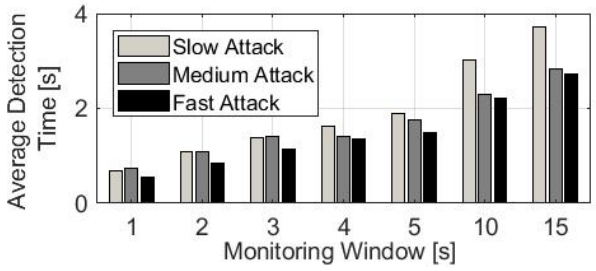
Fig. 9: Traffic behavior when anomaly number of bytes on the same flow entry is detected on the data plane.

plane. Therefore, accurate data plane threshold technically requires more time to be made, as the traffic transmitted between data plane has more variables. Then for the anomaly detection, unlike in the control plane, the counter will compare with threshold at the end of each monitor period, so we can see the counters in 5th and 16th time slot in Figure 9 already surpass their threshold. Then the counter becomes zero after the excess due to the redirection of traffic. From the output in both control and data plane, a threshold from history operation could improve flexibility, and the filter is able to alleviate the impact in the network.

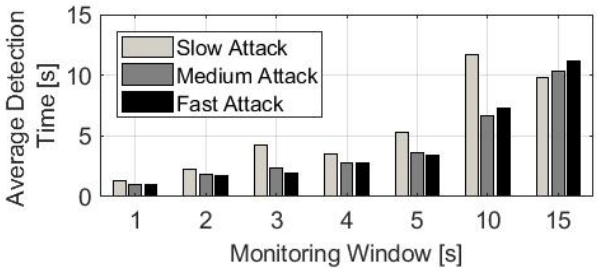
C. Detection Rate and Detection Time

As metrics for evaluated our proposed technique, we use the detection rate (percentage of successful attack detection for both control and data plane), and detection time (time required to detect an attack on control and data plane). In order to verify the performance of proposed model, various monitoring windows β and frequency of attack have been implemented to compare the detection time and rate. The frequency of attack is how fast a malicious packet being sent to the network, either trigger Packet_In messages to the control plane or fake requests to the server in the data plane. We evaluated monitoring window for the following duration {1, 2, 3, 4, 5, 10, 15} seconds with slow, medium and fast attack frequency [17].

Detection time of control plane is given in Figure 10.a with 10 runs for each scenario. We can see from the figure that it usually takes longer for the model to detect slow DDoS attacks, as it requires more time to hit the threshold. By contrast, detection time is shorter if the attack is in a high frequency. In the meanwhile, a big monitoring window results to a long detection time, which is because of the dynamic threshold. Generally, a larger window allows more requests so that the threshold grows bigger as well, and this leads to the fact that it takes more time to hit the threshold. On the data plane in Figure 10.b, since the collection of counter is implemented per monitor window, it is possible that the statistic does not hit its threshold in the first monitor window even the attack has launched, so that DDoS attack detection costs more than a monitor window, which can be seen in the slow attack scenario with 10-second monitor window. Unlike control plane, when DDoS attack is found on the data plane, the counter could already exceed related threshold. As



(a) Control plane.



(b) Data plane.

Fig. 10: Detection time on the control and data plane with various monitor windows.

the monitor window getting larger, this feature increases the unpredictability and instability of detection time, because once it misses detection in the current monitor window, the counter will be reset to 0 and restart counting in the next monitor window. And this is the reason why the trend of bar chart is not always ascending with the same attack frequency.

To show the effectiveness of our algorithm, the result is compared with similar techniques presented in [13], [15] as shown in the Table I. In order to have good accuracy, 210 runs have been tried to validate proposed algorithm on the control and data plane, respectively. In [13], the detection rate is over 96%; While in [15], this value can reach 98%, and detection time is about 2-3 seconds on the data plane depending on the type of attack. From our implementation, the detection rate is 99.9% so far, and detection time ranges from 1 to 11.7 seconds on the data plane, and 0.5-3.7 seconds on the control plane.

TABLE I: Detection rate and time comparison.

Algorithm	Test Rounds	Detection Rate	Detection Time on Data Plane	Detection Time on Control Plane
[13]	9	96.1%	NA	NA
[15]	NA	98%	2-3 s	NA
Proposed	210	99.9%	1-11.7 s	0.5-3.7 s

VII. Conclusions and Future Directions

The combination of SDN and IoT brings convenience in the IoT network management, as well as the challenge to the SDN from the diversity of access device. IoT in Smart city has its own characteristic that the network infrastructure contains a large number of fixed, wired communication nodes, these nodes prefer reliability and security to flexibility. In this paper, we propose to constrain unnecessary routes and register device type so as to reduce complexity in the IoT, in the meantime they raise security level. Behavior-learning based threshold has a broaden application and trustworthy effect in handling various types of attack. Even when a device is hacked, malicious operation will expose the attacker. From the real implementation, both the control and data plane could generate an unique threshold for a device. Based on this threshold, the controller is capable of detect and mitigate suspicious activities.

Acknowledgment

S. Wang is supported by the Australian Government Research Training Program Scholarship.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] A. Whitmore, A. Agarwal, and L. Da Xu, "The Internet of Things a survey of topics and trends," *Information Systems Frontiers*, vol. 17, no. 2, pp. 261–274, 2015.

[3] W. Zhou, Y. Jia, A. Peng, Y. Zhang, and P. Liu, "The effect of iot new features on security and privacy: New threats, existing solutions, and challenges yet to be solved," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1606–1616, April 2019.

[4] S. C. Mukhopadhyay and N. Suryadevara, "Internet of Things: Challenges and opportunities," in *Internet of Things*. Springer, 2014.

[5] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A survey of Software-Defined Networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[6] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on Software-Defined Networking," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 27–51, 2015.

[7] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of Internet of Things for smart home: Challenges and solutions," *Journal of Cleaner Production*, vol. 140, pp. 1454–1464, 2017.

[8] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[9] K. Xu, X. Wang, W. Wei, H. Song, and B. Mao, "Toward software defined smart home," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 116–122, 2016.

[10] O. Flauzac, C. Gonzalez, A. Hachani, and F. Nolot, "SDN based architecture for IoT and improvement of the security," in *IEEE Information Networking and Applications Workshops*, 2015, pp. 688–693.

[11] T. Xu, D. Gao, P. Dong, H. Zhang, C. H. Foh, and H.-C. Chao, "Defending against new-flow attack in SDN-based Internet of Things," *IEEE Access*, vol. 5, pp. 3431–3443, 2017.

[12] A. El-Mougy, M. Ibnkahla, and L. Hegazy, "Software-defined wireless network architectures for the Internet-of-Things," in *IEEE Local Computer Networks Conference Workshops*, 2015, pp. 804–811.

[13] R. Vilalta, R. Ciungu, A. Mayoral, R. Casellas, R. Martinez, D. Pubill, J. Serra, R. Munoz, and C. Verikoukis, "Improving security in Internet of Things with Software Defined Networking," in *IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1–6.

[14] K. Kalkan and S. Zeadally, "Securing Internet of Things (IoT) with software defined networking," *IEEE Communications Magazine*, 2017.

[15] S. S. Bhunia and M. Gurusamy, "Dynamic attack detection and mitigation in IoT using SDN," in *Telecommunication Networks and Applications Conference*, 2017, pp. 1–6.

[16] Open Networking Foundation. (2014) Software-Defined Networking (SDN). [Online]. Available: https://www.opennetworking.org

[17] S. Wang, S. Chandrasekharan, K. Gomez, S. Kandeepan, A. Al-Hourani, M. R. Asghar, G. Russello, and P. Zanna, "SECOD: SDN sEcure control and data plane algorithm for detecting and defending against DoS attacks," in *IEEE/IFIP Net. Operations and Management*, 2018, pp. 1–5.